Publish DHT11 Sensor Data To Adafruit IO Platform using ESP8266

Updated on November 18, 2019

Timothy Malche more

Tim is currently doing research in Internet of Things (IoT). His desire to spread the concept, ideas, and experience of IoT.

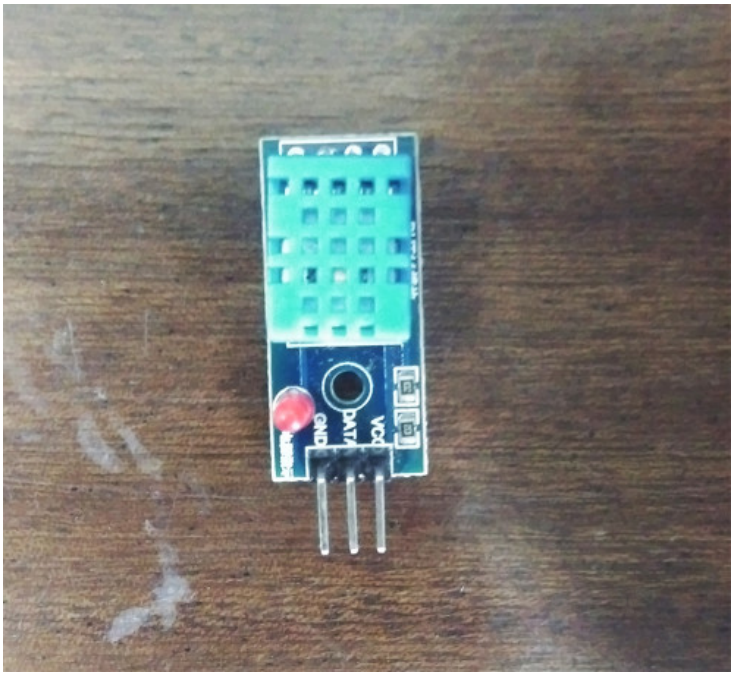
Contact Author

In this tutorial I will show you how send temperature and humidity data from DHT11 sensor to Adafruit IO (AIO) platform via MQTT protocol. I will develop a sensor node which acts as an MQTT client and publishes the data. On AIO platform the published data will be displayed using graphs.

Components Required

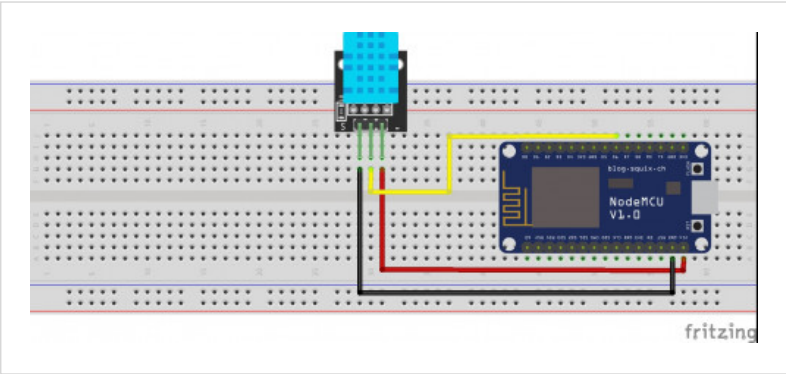
- NodeMCU X 1
- DTH11 Sensor X 1
- Jumper Wires X 3 (Male-To-Male)
- Breadboard X 1

In this tutorial I am using DHT11 sensor which has three pins, DATA, VCC and GND as shown in following figure. You can also use the other DHT11 sensor which has 4 pins.



Circuit

Connect the DATA pin of DHT11 sensor to NodeMCU D6 pin and VCC, GND to Vin and GND pins respectively of NodeMCU. The circuit diagram is shown below.



Arduino Code

Now upload the following Arduino code to NodeMCU. Please remember to specify your WiFi SSID and password and also username and KEY of Adafruit IO (AIO) platform. To get AIO Key follow instruction in my other tutorial.

```
#include <ESP8266WiFi.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"
#include "DHT.h"

/************************* WiFi Access Point *********************************/

#define WLAN_SSID       "YOUR_WIFI_SSID"
#define WLAN_PASS       "YOUR_WIFI_PASSWORD"

/************************* Adafruit.io Setup *********************************/

#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883                   // use 8883 for SSL
#define AIO_USERNAME    "YOUR_AIO_USERNAME"
#define AIO_KEY         "YOUR_AIO_KEY"

/************ DHT11 Setup ******************************************/
#define DHTPIN D6
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

/************ Global State (you don't need to change this!) ******************/

// Create an ESP8266 WiFiClient class to connect to the MQTT server.
WiFiClient client;
// or... use WiFiFlientSecure for SSL
//WiFiClientSecure client;

// Setup the MQTT client class by passing in the WiFi client and MQTT server and login details.
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);

/****************************** Feeds ***************************************/

// Setup a feed called 'photocell' for publishing.
// Notice MQTT paths for AIO follow the form: <username>/feeds/<feedname>
Adafruit_MQTT_Publish temperature = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/temp");
Adafruit_MQTT_Publish humidity = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/hum");


/*************************** Sketch Code ************************************/

// Bug workaround for Arduino 1.6.6, it seems to need a function declaration
// for some reason (only affects ESP8266, likely an arduino-builder bug).
void MQTT_connect();

void setup() {
  Serial.begin(115200);
  dht.begin();
  delay(10);

  Serial.println(F("Adafruit MQTT demo"));

  // Connect to WiFi access point.
  Serial.println(); Serial.println();
  Serial.print("Connecting to ");
  Serial.println(WLAN_SSID);

  WiFi.begin(WLAN_SSID, WLAN_PASS);
  while (WiFi.status() != WL_CONNECTED) {
```

```
    delay(500);
    Serial.print(".");
  }
  Serial.println();

  Serial.println("WiFi connected");
  Serial.println("IP address: "); Serial.println(WiFi.localIP());

}

uint32_t x=0;

void loop() {
  // Ensure the connection to the MQTT server is alive (this will make the first
  // connection and automatically reconnect when disconnected).  See the MQTT_connect
  // function definition further below.
  MQTT_connect();

  // Read humidity
  float h = dht.readHumidity();
  // Read temperature as Celsius
  float t = dht.readTemperature();

  //publish temperature and humidity
  Serial.print(F("\nTemperature: "));
  Serial.print(t);
  Serial.print(F("\nHumidity: "));
  Serial.print(h);

  temperature.publish(t);
  humidity.publish(h);

  delay(60000);

}

// Function to connect and reconnect as necessary to the MQTT server.
// Should be called in the loop function and it will take care if connecting.
void MQTT_connect() {
  int8_t ret;

  // Stop if already connected.
  if (mqtt.connected()) {
    return;
  }

  Serial.print("Connecting to MQTT... ");

  uint8_t retries = 3;
  while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected
       Serial.println(mqtt.connectErrorString(ret));
       Serial.println("Retrying MQTT connection in 5 seconds...");
       mqtt.disconnect();
       delay(5000);  // wait 5 seconds
       retries--;
       if (retries == 0) {
         // basically die and wait for WDT to reset me
         while (1);
       }
  }
  Serial.println("MQTT Connected!");
}
```

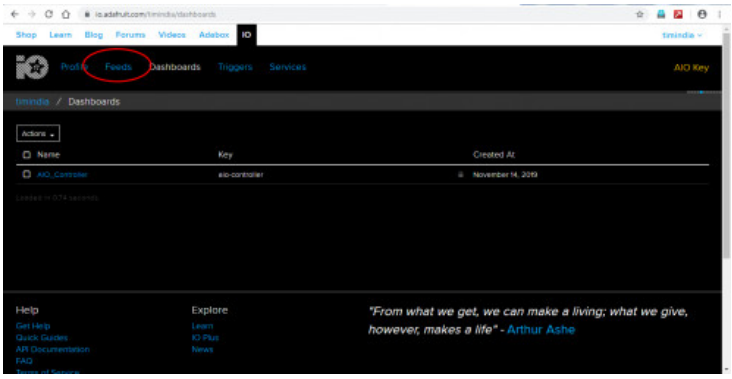
Point to Remember

Please keep noted that I have added 1 minute delay in code (i.e. in line 93) above while publishing the values to AIO platform. Don't upload values very frequently otherwise your account may be blocked for few seconds!

Build User Interface

Our next step is to build the user interface (UI) where we can display the data published by sensor node.

To create UI, logon to your Adafruit IO account.



After you login, you will see AIO_Controller dashboard from previous tutorial. If you have not created the dashboard then please follow the instructions in previous tutorial.

The first thing we need is the Feeds where our data will be published. The Feeds are the MQTT topics. These topics are then bind to our UI which will display data graphically. In this tutorial I'll show another method to create feeds.

For creating feeds goto to **Feeds->View All** as shown in following figure.

You will see already created feeds 'brightness' and 'onoff' from our previous article as shown below. Don't bother if you do not see anything, just create the feeds as instructed in this article.

To do this click on **Actions->Create a New Feed** as shown below.

You will see *Create new feed* dialogbox as shown in following figure. Name the feed as 'temp'.

Repeat the action and create another feed 'hum' as shown below.

Now you will have two feeds 'temp' and 'hum' to store temperature and humidity data as shown below.

Now go to **Dashboard->AIO_Controller**

And click the '+' button to add new block (i.e. Line Chart UI) as shown in following image.

Click on Line Chart UI and select the 'temp' feed from the list of feeds in the dialog box.

Click next and specify block settings

Similarly add another Line Chart block by selecting 'hum' feed as shown in following two images

You will now see dashboard with two line chart blocks.

Now connect your device and you will see temperature and humidity values published on the UI.

Is this article helpful?

Yes

No

Vote See results